(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 January 2002 (03.01.2002)

PCT

(10) International Publication Number
WO 02/01347 A2

(51) International Patent Classification⁷: G06F 9/00

(21) International Application Number: PCT/SE01/01448

(22) International Filing Date: 21 June 2001 (21.06.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/609,111 30 June 2000 (30.06.2000) US

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventors: TSE, Edwin; 4976 Jean Brillant, Montreal, Quebec H3W 1T7 (CA). GOSSELIN, Nicolas; 110 du Blainvillier, Montreal, Quebec J7C 4Y1 (CA). KELLEDY, Fergus; 10 Oriel Terrace, Demense Rd., Dundalk, Co. Louth (IE). O'FLANAGAN, David; 18 Haddington Square, Ballsbridge, Dublin 4 (IE).

(74) Agent: MAGNUSSON, Monica; Ericsson Radio Systems AB, Patent Unit Radio Access, S-164 80 Stockholm (SE).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND SYSTEM FOR AUTOMATIC RE-ASSIGNMENT OF SOFTWARE COMPONENTS OF A FAILED HOST

(57) Abstract: In a network of co-operating hosts (80, 82, 84, 86, 88), a method and system for automatic re-assignment of software components (110, 112) of a failed host to co-operating monitoring (82, 86) or back-up hosts. In a preferred embodiment, a Central Information Repository (CIR), such as an LDAP server, keeps track of software components (110, 112) running on the network hosts (80, 82, 84, 86, 88) and a Monitoring Partnership Program (MPP), in which some hosts (80, 82, 84, 86, 88) monitor the activity of other hosts (80, 82, 84, 86, 88), is provided. Upon failure of a monitored host (84), a monitoring host (82, 86) detects the failure, and informs the other monitoring hosts (82, 86) or the other back-up hosts, if any, of the failure of the monitored host (84). The monitoring hosts (82, 86), and/or the back-up hosts query the CIR for obtaining the identity of the software components (110, 112) running on the failed host (84) before the failure, and select which such components (110, 112) each will start. The monitoring hosts (82, 86) and/or the back-up hosts then take over and start the failed components (110, 112). Upon recovery, the monitored host (84) queries the CIR and obtains the list of its software components, informs the CIR and the monitoring or back-up hosts (82, 86) that it will take over, and starts its components (110, 112), while the monitoring and/or the back-up hosts (82, 86) shut down the components (110, 112) they temporarily run.

METHOD AND SYSTEM FOR AUTOMATIC RE-ASSIGNMENT OF SOFTWARE COMPONENTS OF A FAILED HOST

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to networked and co-operating hosts, and particularly to a method and system for re-assigning a failed host original software components to at least one co-operating host.

Description of the Related Art

Computers have greatly evolved over the last half a century for becoming today a necessity in many areas of technology. Various activities are nowadays exclusively performed by computers, which allows greater and more reliable performance of tasks previously performed by humans.

When different but interrelated tasks are to be processed, one dependable manner of proceeding is to assign specific task(s) to one particular computer and to link a number of computers in a computers' network. According to this type of arrangement, specific software applications may be run on particular computers for performing specific tasks. The computers may be networked, so that the computers' applications can communicate with each other, as initially setup by an operator, for achieving the desired final result.

At the present time, various network configurations exist, each being adapted to a particular type of utilization, such as client-server configuration, chain configuration, cascade configuration, peer-to-peer configuration, federation of co-operating network nodes, etc.

In a network of computers, each computer, or host, may be assigned a number of tasks, i.e. it is only that particular host that performs those tasks. Thereafter, the particular host, connected in a hosts' network, may have to share its processed information with other co-operating hosts. In particular implementations the networked hosts may be linked in "cascade", i.e. each host performs its tasks on the input information received from another host and then outputs the processed information to the next host in the "cascade" network. Failure of only one host in the cascade network results in the overall failure of the network of computers.

Even in other types of arrangements, such as in client-server arrangements, wherein one client's activity depends upon the results of one other client, the failure of one client may result in a total incapacity of achieving the desired global results.

Finally, in practically any type of network configuration, failure of one host that performs tasks which output is essential to the other hosts' operation, may result in critical faults being caused in the overall network, and/or in the incapacity of the hosts' network in accomplishing its global task.

The typical prior art solution to the problem described hereinbefore, is to send a technical operator for manually solve the host failure. Upon detection of an error in a network, various means are typically utilized for locating the problematic host, and a technician takes care for replacing any failed devices, if any, and/or to put the host in normal running condition. However, this solution usually takes significant time, and creates long periods of unavailability (downtime) of the hosts' network.

Another known solution is to have a spare host available, or even a stand-by host for each running host, and once a host failure is noticed in a network, the failed host is replaced with the spare one. Nevertheless, this prior art solution necessitates the existence of at least one "mirror" spare host for each running host, wherein the "mirror" spare host has exactly the same configuration as the running host, thus increasing the costs of hardware and software equipment of the network.

Although there is no prior art solution as the one proposed hereinafter for solving the above-mentioned deficiencies, the US patent 5,729,527 bears some relation with the present invention. In US Patent 5,729,527, Gerstel et al. teach a method and system for rerouting failed channels onto spare channels in a multi-channel transmission system in a networked environment. However, Gerstel et al. are limited to a method and system for solving a link fault and they fail to teach or suggest how to manage a host failure in a network environment.

It would be advantageous to have a method and system for allowing automatic re-distribution of the tasks performed on a particular host in case of the failure of this particular host. It would be of even greater advantage to have a method and system allowing both the re-start of a component after a host failure, and, upon recovery of the failed host, automatic re-insertion of the component at its original logical location in a network of co-operating hosts.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method, a system, a host and a computer-operated software application for monitoring a status of a given host, and upon detection of an unavailability of the given host, to detect the identity of the failed software components that run on the failed host before the failure, and to re-start the mentioned components on another host(s).

According to the invention, there is provided a group of co-operating hosts, wherein at least one monitoring host monitors the activity of at least one monitored host. Upon detection of a failure of the monitored host, the monitoring host informs a Central Information Repository (CIR) of the failure of the monitored host. The CIR, that may be physically a distributed database but is preferably logically centralized, further informs at least one back-up host, that may be another monitoring host, and the components that failed on the monitored host are re-started on the back-up host. Preferably, the back-up hosts may be the same as the monitoring hosts, and in this particular case the failed components are restarted on the monitoring hosts. Accordingly, it is an object of the present invention to provide in a network of co-operating hosts, a method for re-assigning at least one software component of a monitored host to one or more back-up hosts if said monitored host experiences a failure, the method comprising the steps of:

detecting a failure in the monitored host;

determining at least one component that was running on said monitored host before the failure; and

starting a copy of said at least one component on said one or more back-up hosts, wherein each copy of said at least one component is started and run on a given one of said one or more back-up hosts.

It is another object of the invention to provide in a network of co-operating hosts, a host comprising:

a Component Manager (CM) for managing local components and for monitoring an activity of at least one of said co-operating hosts;

wherein upon detection of a failure of one of said at least one of said co-operating hosts running a given component, said CM starts and runs a copy of said given component.

It is yet another object of the invention to provide a network of co-operating hosts comprising:

a monitored host running at least one software component;

one or more monitoring hosts for monitoring an activity of said monitored host;

one or more back-up hosts, each one of said back-up hosts comprising a Component Manager (CM), and at least one installed component;

wherein when a failure occurs in said monitored host, said one or more monitoring hosts detect said failure and start said at least one software component on at least one of said back-up hosts.

It is yet another object of the present invention to provide in a network of co-operating hosts, a method for re-assigning each software component of a monitored host to one or more monitoring hosts if said monitored host experiences a failure, the method comprising the steps of:

detecting a failure in the monitored host by a first monitoring host;

notifying a Central Information Repository (CIR) of the failure of the monitored host by said first monitoring host;

verifying, in said CIR, if other monitoring hosts than said first monitoring host are also responsible for monitoring an activity of said monitored host;

if other monitoring hosts are responsible for monitoring said activity of said monitored host, informing said other monitoring hosts of the failure of the monitored host;

obtaining, for each one of said first monitoring host and said other monitoring hosts, a list of software components run prior to said failure by the monitored host;

dividing a responsibility of re-starting individual components of said list among each one of said monitoring hosts; and

starting and activating each one of said individual components on a selected monitoring host according to said division of responsibility.

It is yet another object of the invention to provide a computer-operated software application for managing local software components running on a local host and for monitoring an activity of at least one networked monitored host, said application including a local Component Manager (CM) comprising:

means for detecting a failed monitored host;

means for obtaining an identity of at least one component run by said monitored host; and

means for starting and running a copy of said at least one component, wherein at least part of said copy is installed on said local host.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more detailed understanding of the invention, for further objects and advantages thereof, reference can now be made to the following description, taken in conjunction with the accompanying drawings, in which:

Figure 1.a is a top level block diagram of a network of co-operating hosts according to an exemplary prior art implementation;

Figure 1.b is a top level block diagram of an Event Management System (EMS) according to an exemplary prior art implementation;

Figures 2 (a, b, and c) are high-level block diagram illustrating an exemplary preferred embodiment of the invention;

Figure 3 is a nodal operation and signal flow diagram illustrating an exemplary preferred embodiment of the invention;

Figure 4 is a high-level flowchart of another exemplary preferred embodiment of the invention; and

Figure 5 is a nodal operation and signal flow diagram illustrating yet another exemplary preferred embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Reference is now made to Figure 1.a, wherein there is shown is a top level block diagram of a network 10 of co-operating hosts according to an exemplary prior art implementation. Hosts 12, 14, 16 and 18 (A, B, C and D) are linked through a network 20. It is understood for the purpose of the present example that all hosts 12-18 comprise the necessary network interfaces (not shown), i.e. network card, network connections, network software applications, etc, that allow each one to be appropriately connected to each other, through the network 20. Each host comprises an Operating System (OS), not shown, which supports various software applications, hereinafter called Components. In Figure 1.a, for example, Host A, in its Enabled (up and running) state runs three components C_{A1}, C_{A2}, and C_{A3}, Host B runs another three components C_{B1}, C_{B2}, and C_{B3}, Host C runs two components C_{C1}, and C_{C2}, while Host D runs a single component C_{D1}. In the present example the illustrated components are all part of a distributed application which runs onto the different hosts 12-18. Therefore, the activity of the components is interrelated, some components activity being dependent upon other components proper output. In the present example, it is further assumed that component C_{B3} of Host 14 (B) transforms its input 22 (received from one other component of one other host via network 20) and sends its output information 24, through the network 20 to component C_{C1} of Host 16 (C) which further processes information 24. In the case of a failure of Host 14 (B), the component C_{B3} becomes unavailable (is down) and is thus no longer capable of achieving its task of outputting the information 24 to component C_{C1}. Therefore, in such a case, the processing chain is broken, and the distributed application no longer achieves its global task. This drawback of the prior art implementation, wherein failure of one, or of a few components, result in failure of the overall distributed application, may also occur in other types of networks.

Figure 1.b is a top level block diagram of an Event Management System (EMS) 30 in charge of monitoring a network 32. The monitored network 32 may by any kind of network, such as for example a Local Area Network (LAN) over Ethernet, an Internet Protocol (IP) network, or a Public Local Mobile Network (PLMN). Typically, each network may have an associated EMS which is used by the network operator in charge of that network to monitor the network activity. The typical tasks of an EMS are to collect the events originating from the monitored network 32, to process the events (conversion, treatment, classification, etc.), to store the events and finally to display the events, in particular formats, onto network administrators' monitoring means. In the particular example shown in Figure 1.b, the Host 34 runs a component 36 dedicated to collecting (trapping) the events issued by the monitored network 32 via the Gateway 33. The component 36 traps the events and further outputs the events flow 38 to host 40 running component 42 which is dedicated to converting the incoming flow of events 38 into a user-friendly formatted events flow, 44. The information 44 is then sent to host 46 running a database component 48 for storing the event-related information 44. Finally, hosts 50, 52, 54, and 56 respectively run individual components (not shown) which are dedicated to the display of the event-related information 48, which may be accessed on a by-request basis. Those skilled in art would readily notice that the particular example of Figure 2.a shows another distributed chain application, wherein different hosts are chain-connected for achieving one global task of event monitoring. Again, if one particular host fails (for various reasons such as power outage, crashed OS, memory fatal corruption, etc.), the global task of the EMS 30 is interrupted.

One partial remedy known in the prior art to the aforementioned problem is to "duplicate" a host with a "mirror" having identical configuration. For example, shown in Figure 1.b, is host 34 running component 36 that is dedicated to collecting the events from the network 32. A "mirror" host 34' running the same component as host 34, namely 36', may be incorporated in the EMS 30 and run in stand-by mode. If a failure is detected in host 34, then the stand-by host 34' takes over and assumes the tasks of the failed host 34. However, this solution implies duplicating each host of the network, thus doubling the costs of hardware equipment, while half of this equipment (the stand-by hosts) is only used in critical situations.

Reference is now made to Figure 2, which shows a high-level block diagram of an exemplary preferred embodiment of the present invention. According to a broad aspect of the invention, there is provided a method and system for automatically re-assigning the components running on a particular host to a co-operating networked host (i.e. that exchange information in at least one direction), upon detection of a failure of the particular host.

In Figure 2, Hosts 80, 82, 84, 86 and 88 (A-E) are all connected to a network (not shown) and function in co-operation with each other. Each one of the hosts 80-88 runs at least one component, such as for example component 90 (C_{B1}) for host 82 (B), that is typically a software application responsible for performing one or more particular tasks. The components running on the various hosts 80-88, may be in quasi-permanent communication with each other, in a by-request communication, or in any other known type of communication wherein information must be transmitted from one host to another.

According to a preferred embodiment of the present invention, a Monitoring Partnership Program (MPP) is implemented among at least two networked hosts that co-operate for achieving a global task. According to the MPP, the participating hosts reciprocally monitor each other's activity and, upon detection of a fault, error, malfunction or other unavailability of a particular host, the fault or the like of a monitored host is detected by a co-operating monitoring host, the unavailable components that were running before the occurrence of the fault on the monitored host are detected as well and are started onto the partner monitoring host(s).

In Figure 2.a, there is shown an exemplary high-level block diagram of a hosts' network in its normal operation wherein five different hosts (80-88) run various components C_{Xi} that inter-communicate with each other. For the purpose of the example, it is assumed that only the hosts 82, 84, and 86 participate to the MPP. Thus, for example, the activity of the monitored host 84 (C) is supervised by both the monitoring host 82 (B) and the monitoring host 86 (D). Each one of these co-operating hosts 80-88 comprises, besides its running components, a

Component Manager (CM), 100, 102, 104, 106, and 108, which may be a software application responsible for launching (at start-up) and monitoring (during regular operation) the hosts components. Furthermore, it is also the CMs of each host that may be responsible of the MPP for its corresponding monitored host(s), i.e. the monitoring of the partner hosts. For example, in Figure 2.a the CMs 102 and 106 of the monitoring hosts 82 (B) and 86 (D) are responsible for monitoring the activity of the host 84 (C), by sending for example, a request for a heartbeat signal (not shown) to host C. According to the MPP, each monitoring host, such as host 86 (D), also comprises a Library of Components (LC) 101 comprising the components C_{Xi} 103_i running on the monitored hosts (such as the components 110 and 112 of the monitored host 84 (C)). The LC 101 may be the same for all the network co-operating hosts, in which case it comprises the installed components 103_i of all the co-operating hosts running in the network, or may be unique for each monitoring host, in which case it may comprise, besides the components naturally running on the particular host, only components 103_i that are running on the host(s) monitored by the monitoring host. It is to be understood that although the CL 101 and the installed components 103_i are only represented for the monitoring host 86 (D), all the monitoring hosts, or even all the hosts may comprise such an LC 101. Furthermore, LC list 101 must not necessarily comprise the full version of the installed components 103_i, but may alternatively comprise only a portion thereof that, when activated, can automatically contact a central server for performing the full download, and start of a particular component.

In Figure 2.b, host 84 fails. This may be caused by various sorts of problems, such as a power outage, a physical accident, a memory corruption, a crush of the OS or others. According to the MPP, the CMs 102 and 106 of hosts 82 (B) and 86 (D) actually monitor, continuously or from time to time, the activity of host 84 (C), and thus become aware of the failure of host 84 (C). Thereafter, the co-operating CMs 102 and 106 may inquire of the identity of the components that were running on the host 84 (C) before the failure. This action (request) may be addressed to a Central Information Repository (CIR, not shown), such as to an LDAP server, that has knowledge of the network topology, and of the particular components assigned and run onto each host. Alternatively, the request for the identity of the failed components may be performed toward any one of the hosts that may have knowledge of the network-related information, or purely skipped if each hosts has knowledge of the network-related information. Subsequently, once the co-operating CMs 102 and 106 of the monitoring hosts have knowledge of the identity of the components of host 84 (C), they may divide the responsibility of starting and running the failed components 110 and 112 according to a pre-defined scheme that will be discussed later in this document. It is understood that if only one host, such as only host 86 (D) is set-up to monitor the activity of host 84 (C), then the step of dividing the responsibility is skipped or, alternatively and preferably still performed, with the result, in both cases, that the responsibility of starting and running the failed components will be assigned to the single monitoring host.

Thereafter, once the responsibility of starting and running the failed components is divided, an equivalent copy of the components that failed on host 84 (C) is started, by choosing the right component from the components 103_i of the library 101, and run on the monitoring hosts 82 (B) and 86 (C). The "displaced" components 110' and 112' of Figure 2.b are responsible for re-inserting themselves at their original logical location by making the required original logical connections 89', 91', 93', 95', and 97' and synchronization.. For example, the components 110 and 112 were initially running on host 84 (C), as shown in Figure 2.a, at their original logical location. When host 84 (C) fails, as shown in Figure 2.b, the components 110 and 112 are "displaced" on host 82 and 86 respectively, i.e. once they become unavailable on host 84 (C), such as for example because host 84 (C) failed. Their equivalents (selected components from library 101) are launched on host 82 and 86 (D) as new running components 110' and 112', by the CMs 102 and 106. It is understood that for achieving the invention, each host participating to the MPP must comprise, or alternatively have access to copies of the components (software applications 103_i) of its co-operating monitored hosts. For example, host 82 (B) may comprise among its installed components 103_i the component 110', which is started and run in the present example upon detection of the failure of host 84 (C). Alternatively, the back-up (monitoring) hosts participating to the MPP may i) contact the CIR 140 upon detection of a failed monitored host, download from the CIR 140 the required components that need to be re-started, and re-start these components, or, ii) may comprise a portion of the copy of the failed component(s), that may be activated upon detection of the failure of the original components, and that may further take care of the full download from the CIR 140 of the remaining portion of the component copies, which are then re-started on the back-up (monitoring) hosts.

Reference is now made to Figure 2.c, wherein there is shown the scenario of the recovery of host 84 (C). At a later point in time, it is assumed that the problem that caused the failure of the host 84 (C) is corrected, and that host 84 (C) recovers its Enabled state. At this stage, the recovery of the host 84 (C) is detected by the monitoring hosts according to a scheme to be farther discussed, the substitute components 110' and 112' are stopped by the CMs 102 and 106, and the original components 110 and 112 (i.e. their respective original copies) are started on host 84 (C) by the CM 104. The newly started components 110 and 112 of Figure 2.c are responsible for re-inserting themselves at their original logical location by making the required original logical connections 89, 91, 93, 95, and 97 and synchronization. This may be achieved by providing the components with information regarding with which other component it must communicate or alternatively and preferably, the newly started components can get this information from the CIR 140.

Reference is now made to Figure 3, which is a nodal operation and signal flow diagram of an exemplary preferred embodiment of the invention showing a possible actual implementation of an MPP with three hosts, wherein the activity of the (monitored) host 84 (C) is set to be supervised by the co-operating (monitoring) hosts 82 (B) and 86 (D). It is to be understood that although a simplified MPP is described in the forthcoming description, other combinations may exist according to the invention between the monitoring and the monitored hosts, such as but not limited to one monitoring host for one monitored host, one monitoring host for a plurality of monitored host, etc. With reference back to the example of Figure 3, each host 82-86 comprises a Component Manager (CM) 102-106, responsible for managing the Components running on the respective host. In the exemplary preferred embodiment illustrated in Figure 3, the CM 102 of host 82 (B) controls the running components 120 and 122, the CM 104 of host 84 (C) controls the running components 124, 126 and 128, while the CM 106 of host 86 (D) controls the running components 130 and 132. Furthermore, the network also comprises a Central Information Repository (CIR) 140, which is preferably a centralized or distributed LDAP server, that may contain a Component List (CL) 142 and a Component Manager List (CML) 144. Preferably, the CL 142 comprises a plurality of Component records 146_i containing information about the hosts' components, each component record having a field Preferred Host Name $146_{i\text{-PHN}}$ for holding the identity of the host naturally running the component, and a field Actual Host Name $146_{i\text{-AHN}}$ for holding the identity of the actual host running the component (in case of unavailability of the preferred host). The CML 144 preferably comprises a record 148_i for each CM 102-106, and each record 148_i further comprises a field Monitored Hosts $148_{i\text{-MH}}$ for holding the identity of the hosts monitored by each CM according to the MPP, and a field Operation State Attribute $148_{i\text{-OSA}}$ for holding the status of the host's CM, such as "Enabled" when one particular host and its CM is up and running, or "Disabled" when the particular host and its CM is down or otherwise not available.

At step 150 a critical error occurs in host 84 (C) such that the host becomes unavailable. The host 84 (C) can alternatively become unavailable for any other reason. According to the MPP, the partner hosts 82 (B) and 86 (D) monitor the activity of host 84 (C). This may be achieved for example by regularly sending a heartbeat request signal 152 from the monitoring hosts 82 and 86 to the monitored host 84 (C). Upon receipt of the heartbeat request signal 152 assumed to be sent by host 82 (B), if the host 84 (C) were to be enabled (i.e. up and running), it would have sent back to host 82 (B) a heartbeat response signal for acknowledging the fact that it is Enabled and running components 124, 126, and 128. However, in the present case, the host 84 (C) is unavailable and the heartbeat response signals is not sent back to host 82 (B). At step 154, host 82 (B) detects the absence of heartbeat response signal (ex.: timer timeout) and deduces that the host C is unavailable. It is to be noted that the detection of the unavailability of host 84 (C) can be also detected by other particular signaling implementations. For example, signal 152 may be skipped and host 84 may be set-up to regularly signal its Enabled state to its co-operating hosts according to the MPP. Failure to do so would result in the conclusion for its partner monitoring host(s) that the monitored host is unavailable. Other error messages may be used as well.

After the first detection of the unavailability of host 84 (C), action 154, a notification of unavailability 156 is sent from the CM 102 of host 82 to the CIR 140. The notification 156 may comprise the new state "Disabled" of the host 84 (C) or any other indication that host 84 (C) is now unavailable. Upon receipt of the notification 156, the CIR 140 modifies the operational state attribute field $148_{i\text{-OSA}}$ of the CM 148_i corresponding to host 84 (C), from "Enabled" to "Disabled", action 158, in order to reflect the unavailable state of host 84 (C). Thereafter, the CIR 140 locates in the CML 144, using the field Monitored Hosts $148_{i\text{-MH}}$ of the records 148_i if other CMs of other hosts, are responsible of the failed host 84 (C). In the present example, besides the host 82 (B) that first detected the failure of host 84 (C), host 86 (D) is detected in action 160 as being also responsible of the failed host 84 (C). The CIR 140 further informs the CM 106 of host 86 that host 84 (C) became unavailable, by sending an indication, action 162. It is to be noted that in the particular scenario wherein only one monitoring host is responsible of a monitored host that failed, action 160 returns no other CM's identity and therefore action 162 is skipped.

At this stage each one of the monitoring CMs 102 and 106 running on hosts 82 (B) and 86 (D) are aware that host 84 (C) is unavailable, and in actions 164 and 166 they query the CIR 140 for the identity of the failed components (124, 126 and 128) that run on host 84 (C) before the failure, by sending a request for components identity along with the host C identity. Upon receipt of the queries 164 and 166, the CIR uses the host C identity for extracting from the CL 142 each component identity whose Actual Host Name entry of field $146_{i\text{-AHN}}$ matches the identity of host 84 (C), action 168, and returns this information (a list of components, 173) to the CMS 102 and 106 of the monitoring hosts 82 (B) and 86 (C) in actions 171 and 172. Alternatively, action 168 can be separately performed for example twice, after individual receipt of messages 164 and 166. In action 174, the CMs 102 and 106 select which components (from the components list, 173) each one is to take care of, and take also the responsibility of monitoring the hosts previously monitored by host 84 (C), in a manner that is yet to be described.

For the purpose of the example illustrated in Figure 3, it is assumed that following action 174, the CM 102 of the host 82 (B) is assigned the responsibility of starting and running components 124 and 126, while the responsibility of starting and running component 128 is assigned to the CM 106 of host 86 (D). Furthermore, according to a preferred embodiment of the invention, the monitoring hosts 82 (B) and 86 (D) are also the ones that inherit, after the failure of the monitored host 84 (C) of the responsibility of monitoring the hosts previously monitored by host 84 (C).

Therefore, in action 176, the CM 102 starts the installed components 103_i corresponding to the failed components 124 and 126 that becomes the running components 124' and 126', not shown, which are copies of software applications identical to components 124 and 126 that became unavailable on host 84 (C), with the difference that they are launched on host 82 (B). In an analogous action 178, the CM 106 starts the installed component 103_i that corresponds to the failed component 128, that becomes the components 128', which is the same software application as component 128, with the difference that it is launched on host 86 (D). Finally, each newly launched component 124', 126' on host 82 (B) and component 128' on host 86 (D) is activated in actions 180 and 182 respectively, by establishing the logical connections with other co-operating components from within the same host, or from the other networked hosts. Alternatively, the started components themselves may have the responsibility and the capacity of establishing the logical connections with their respective cooperating components. This may be achieved, for example, by including in the components to be started information comprising the identity of their cooperating components, and/or information related to the logical path the communications should follow, or by setting the newly started components to contact the CIR 140 for retrieving the information relating to the identity of their cooperating components.

Reference is now made to Figure 4 wherein there is shown a high-level operational flowchart diagram of the action 174 (from Figure 2) for determining i) mainly the division of the responsibility for starting and running the components 124, 126 and 128 of the failed host 84 (C) between the partner monitoring hosts 82 (B) and 86 (C), and ii) also the takeover of the responsibility of monitoring the hosts previously monitored by host 84 (C) by the monitoring hosts 82 (B) and 86 (D).

When one or more components become unavailable because the failure of a host, such as the one described for host 84 (C), and when a plurality of monitoring hosts share the responsibility of supervising that host, a decision must be taken regarding the manner in which the failed components will be re-started by the monitoring hosts, i.e. which host will re-start and run which component. The decisional sequence corresponding to this decision, action 174 of Figure 3, is herein described with reference to Figure 4.

Upon receipt by the CM 102 of the monitoring host 82 (B) of the component list 173 (the same procedure applies to host 86 (D) as well but for simplification purposes will be only described in relation to the host 82 (B)), the CM 102 of host 82 (B) takes over the responsibility of monitoring the hosts previously monitored by host 84 (C). This may be achieved by updating the record $148_{i\text{-}MH}$ of the monitoring host 82 (B) for further including the hosts previously monitored by host 84 (C). Thereafter, the CM 102 of host 82 (B) selects one component from the list 173, such as for example the first component (component 124) from the list, action 200. Alternatively, the selection of the components from the list 173 may be made randomly, or according to other logic as believed appropriate and implemented by the network operator.

The CIR 140 is then queried and the Actual Host Name entry corresponding to the selected component 124 is obtained, action 202. At the same time, a Lock Record Action is performed for this particular component's record 146_i in the CIR 140. Thereafter, the Actual Host Name Entry obtained in action

202 is compared with the identity of Host C (previously obtained in action 154 or action 162), action 204. If the comparison is a perfect match, i.e. the Actual Host Name Entry obtained in action 202 is the same with the identity of the failed monitored Host C, it is deduced that in the meantime no other monitoring host (such as the partner monitoring host 86 (D)) has already taken charge of this component, and therefore an update is performed, action 206, from the monitoring host 82 (B) to the CIR 140, for changing the Actual Host Name entry in the field $146_{i\text{-AHN}}$ of the record 146_i with the host name of the monitoring host 82 (A) in order to reflect that host 82 (A) is about to take care for re-starting and running the selected component 124. Action 206 may comprise an update request being sent from the CM 102 to the CIR 140, the actual update at the CIR and an update acknowledge being sent back from the CIR to the CM 102. Thereafter, the CM 102 requests a Lock Release for the record 146_i, and the CIR releases the Lock, action 208. Afterwards, the CM 102 deletes the selected component from the list of remaining components 173, action 209, and writes or keeps in its memory the selected component identity, action 210, that allows it to continue with subsequent actions (176 and 180) shown in Figure 3 and described beforehand for this particular component. It is to be noted that the order of actions 209 and 210 can be inverted.

With reference being made back to action 204 of Figure 4, if the result of the comparison is not a perfect match, i.e. if the Actual Host Name Entry obtained in action 202 is not the same with the identity of the failed monitored Host C, it is concluded that another host, such as host 86 (D) took charge of re-starting and running the selected component 124, and at the same time wrote its own identity in the Actual Host Entry field $146_{i\text{-AHN}}$ of the selected component record 146_i. In this case, the CM 102 requests and obtains a Release Lock of the record 146_i corresponding to the Component 124, action 212, and further deletes the selected component from the list of remaining components to be considered. At the end, both after action 210 and action 213, the process restarts for each one of the remaining components, such as for components 126 and 128.

The sequence of actions described hereinbefore in connection with Figure

4 is repeated for each component received in the component list 173 by the CM 102, so that the CM 102 of host 82 (B) attempts to take charge of each one of the components 124, 126, and 128, and succeeds in doing so only for the components that were not yet taken in charge by the other monitoring host 86 (D). Furthermore, substantially at the same time, the same repetitive sequence is performed by the CM 106 of host 86 (D) with respect to the same failed components 124, 126, and 128 thus resulting in the partition of the responsibility for re-starting and running these components by the two monitoring hosts 82 (B) and 86 (D). It is to be understood that according to the proposed scheme, "partition" may also mean that one host get no responsibility for any of the components while the other host can get the responsibility of all the failed components.

In an alternative embodiment of the invention, the failed components selection of action 174 may be performed according to a pre-determined arrangement wherein particular hosts can automatically be assigned the responsibility of certain failed components. For example, it could be pre-determined that in case of failure of host 84 (C), components 124 and 126 would be re-assigned to host 82 (B) while component 128 would be assigned to host 86 (D) without performing the decisional sequence of Figure 4. This pre-determined information may be stored in the CIR 140 and transmitted to the monitoring hosts, or in the monitoring hosts 82 (B) and 86 (D) themselves. In this later case, wherein the monitoring hosts have knowledge of the components that were running on the host 84 (C) and of the potential partition of these components in case of failure of host 84 (C), the actions 164-172 of Figure 3, and the actions 200-212 of Figure 4 may be skipped.

Reference is now made to Figure 5, which is a nodal operation and signal flow diagram showing the sequence of actions performed upon recovery of the host 84 (C). In action 300, host 84 (C) recovers after a period of unavailability, and its CM 104 starts and becomes Enabled and running. Once the host 84 (C) is "up and running" its CM 104 becomes aware that there are no components running and, following action 300, it signals the CIR 140 and queries for the identity of the components it should be running, action 302. This may be achieved by sending its host identity along with a request for components. Upon receipt of the query, the CIR 140 may extract from the CL 142 the identity of the components that would preferably run on host 84 (C), action 303, by consulting for example the Preferred Host name field $146_{i\text{-PHN}}$ of the component records 146_i of list 142. All the components whose entry of field $146_{i\text{-PHN}}$ matches the host 84 (C) identity are returned in a component list 304 of components that would be preferably run on host 84 (C), in action 306. Upon receipt of the list 304, the host 84 (C) starts each of the components of the list 304 one at a time, such as for example component 124 in action 308, and for each such component performs the following actions. Once the component is started (ex.: component 104 is started), in action 310 the CM 104 sends a request for update of the list 142 to the CIR 140, by including in the request the component identity (ex. Component 124's identity). In action 312 a Lock Record is performed on the record 146_i of the component 124 and the entry of the field $146_{i\text{-AHN}}$ is read, action 314. Thereafter, the Actual Host Name entry read in action 314 is returned to the host 84 (C) in action 316. In the present example, since it was host 82 (B) that temporarily took charge of component 124, it is host B' identity that is returned in action 316. In action 318 it is determined if the actual host name entry returned is different from host 84 (C) own identity, and if yes, the host corresponding to the Actual Host name entry returned (host 82 (B)) is signaled with a Component Shutdown Request 320 for the component 124, to which the host 82 (B) responds by first, shutting down the component 124' (the host 82 (B) equivalent of component 124), action 322, second, by stopping monitoring the activity of the hosts that were to be monitored by host 84 (C) (which implies an update of the record $148i_{\text{-MH}}$ of the CML 144 in the CIR 140, i.e. the deletion of the identity of the hosts originally monitored by host 84 (C)), action 324, and third, by sending back a Release Lock Acknowledge message 326. Upon receipt of the message 326 that confirms that host 82 (B) shut down the component 104, the CM 104 of the host 84 (C) sends an Update Actual Host Name request message, action 328, to the CIR 140 for requesting the change of the record 146_i, particularly of the field $146i_{\text{-AHN}}$ corresponding to the component

124, from host 82 (B) to host 84 (C) in order to reflect that host 84 (C) took over the responsibility for running the selected component. Upon receipt of the Update Actual Host Name request message, the CIR 140 proceeds to the update of field 146_i, action 330, releases the lock on the record field 146_i, action 332, and returns back to the CM 104 of the host 84 (C) an update acknowledgement, action 334. The CM 104 then initiates the activation of the component 124, action 336, i.e. the component 124 proceeds to its insertion at its natural logical location by communicating with its co-operating components and by establishing the required connections. In a variant of the preferred embodiment of the invention, action 336 may also comprise a certain synchronization of the data status of the newly started component 124 with the component 124'. For example, the old data status of component 124' running on host 82 (B) may be read in action 322 before shutting down the component 124', and may be transmitted to the CM 104 of host 84 (C) in action 326 along with the Release acknowledge message. Subsequently, the CM 104 of host 84 (C) may use the old data status read from component 124' for synchronizing the newly started component 124, i.e. the old data status read from component 124' would become the new data status of the newly started component 124.

Reference is now made back to action 318, wherein it is determined if the actual host name entry returned in action 316 is different with respect to host 84 (C) own identity. Actions 320-334 are preferably only performed if the entry in the filed $146_{i\text{-}AHN}$ (the actual host name entry returned in action 316) is different from the host 84 (C) identity, i.e. only if another monitoring host did actually temporarily took charge of the component 124 (exceptions may occur such as for example in the case of a resources overload of the monitoring host). In the exception case wherein the entry in the filed $146_{i\text{-}AHN}$ (the actual host name entry returned in action 316) is the same as the host 84 (C) identity, the actions 320-334 are therefore preferably skipped.

In yet another embodiment of the invention described with reference to Figure 2, at least one of, or both the hosts 82 (B) and 86 (D) may not be monitoring hosts, but rather only assume the function of re-starting the failed components of the monitored host 84 (C). Thus, the function of monitoring the status of the monitored host 84 (C) may be assign to one or more host(s) different from the host(s) whose function is to re-start the failed component(s). For example, in Figure 2, host 80 (A) may be the monitoring host of host 84 (C), and may first detect the failure of the monitored host 84 (C). Thereafter, it is the responsible (back-up) hosts 82 (B) and 86 (D) that have the responsibility of re-starting and running the failed components of the monitored host as described hereinbefore with reference to Figures 3, 4 and 5, with the exception that the monitoring host that first detects the failure of the monitored host, action 154, may be different from the hosts 82 (B) and 86 (D) that actually re-start the failed components 124', 126', and 128', actions 176-182. According to this preferred embodiment of the invention, the back-up hosts may be any type of co-operating host that has installed copies of the software components of its corresponding monitored host, or have access to these copies such as for example from the CIR 140.

It is to be noted that the invention as described hereinbefore can be implemented in various forms, as best suited in a particular network. In particular, the CIR 140 can be any type of unified or distributed database application, such as for example a centralized or distributed LDAP server. In this former case in which the CIR 140 is an LDAP server, advantage can be obtained from the particular functionalities of LDAP. For example, some notifications, such as but not limited to actions 160 and 162 can be automated by placing a "notification request upon change" request in the LDAP sever regarding the Operational State Attribute of Host C.

Although several preferred embodiments of the method and system of the present invention have been illustrated in the accompanying Drawings and described in the foregoing Detailed Description, it will be understood that the invention is not limited to the embodiments disclosed, but is capable of numerous rearrangements, modifications and substitutions without departing from the spirit of the invention as set forth and defined by the following claims.

WHAT IS CLAIMED IS:

1. In a network of co-operating hosts, a method for re-assigning at least one software component of a monitored host to one or more back-up hosts if said monitored host experiences a failure, the method comprising the steps of:

detecting a failure in the monitored host;

determining at least one component that was running on said monitored host before the failure; and

starting a copy of said at least one component on said one or more back-up hosts,

wherein each copy of said at least one component is started and run on a given one of said one or more back-up hosts.

2. The method claimed in claim 1, wherein said back-up hosts are monitoring hosts also responsible for monitoring a status of said monitored host.

3. The method claimed in claim 2, wherein said step of detecting a failure is performed by at least one of said one or more monitoring hosts and comprises one of:

i) an absence of heartbeat response of said monitored host to at least one of said monitoring hosts; and

ii) an error message sent from said monitored host to said one of said monitoring hosts.

4. The method claimed in claim 2, wherein the step of detecting at least one component that was running on said monitored host comprises the steps of:

querying by each one of said monitoring hosts a Central Information Repository (CIR) for an identity of said at least one component that was running on said monitored host; and hosts an identity of said at least one component.

5. The method claimed in claim 2, wherein

said one or more monitoring hosts comprise a plurality of monitoring hosts;

the at least one component comprises a plurality of components;

the step of determining comprises receiving at each one of said monitoring hosts a list of said plurality of components from a Central Information repository (CIR); and

the method further comprises, prior to the step of starting, the steps of:

for each component of said plurality of components, selecting one monitoring host from said plurality of monitoring hosts for starting said individual component, and subsequent to the step of selecting, updating said CIR with a name of said one monitoring host, whereby the update of the CIR is made in order to reflect that said one monitoring host took charge for starting and running a particular component.

6. The method claimed in claim 2, further comprising, subsequent to the step of starting, the steps of:

recovering the monitored host;

obtaining a list of said at least one component that was running on said monitored host before the failure;

re-starting said at least one component on said monitored host; and

shutting down said copy of said at least one component on said one or more monitoring hosts.

7. The method claimed in claim 6, wherein the step of obtaining comprises the steps of:

querying by the recovered monitored host the CIR for obtaining said list; and

obtaining from the CIR said list.

8. The method claimed in claim 6, wherein the step of re-starting comprises the steps of:

re-starting each component received in said list; and

informing the CIR that said recovered monitored host took over a responsibility for starting and running said least one component of said list.

9. The method claimed in claim 6, wherein the step of shutting down comprises the steps of:

the recovered monitored host obtaining from said CIR an identity of a first monitoring host running a copy of one of said at least one component;

signaling by the monitored host said first monitoring host for requesting the shutdown of said copy; and

shutting down by said first monitoring host of said copy.

10. The method claimed in claim 8, further comprising the step of:

activating each one of said at least one component of said list on the recovered monitoring host; and

synchronizing each one of said at least one component of said list.

11. In a network of co-operating hosts, a host comprising:

a Component Manager (CM) for managing local components and for monitoring an activity of at least one of said co-operating hosts;

wherein upon detection of a failure of one of said at least one of said co-operating hosts running a given component, said CM starts and runs a copy of said given component.

12. The host claimed in claim 11, wherein before starting said copy of said given component, said CM first signals a Central Information Repository (CIR) for informing that it takes over a responsibility of starting and running said given component.

13. The host claimed in claim 11, said host further comprising at least one running component for achieving a particular task dedicated to said host.

14. The host claimed in claim 11, wherein said host is a monitoring host and further comprises a Library of Components (LC) having information related to a series of components run by a number of co-operating hosts from said network.

15. The host claimed in claim 14, wherein said LC is unique to a number of co-operating monitoring hosts including said monitoring host, and said information relates to a series of components run by a number of monitored hosts monitored by said monitoring hosts.

16. The host claimed in claim 14, wherein said information comprises a series of installed components corresponding to said series of components run by a number of co-operating hosts from said network.

17. The host claimed in claim 14, wherein said information comprises a series of partially installed components corresponding to said series of components run by a number of co-operating hosts from said network.

18. A network of co-operating hosts comprising:

a monitored host running at least one software component;

one or more monitoring hosts for monitoring an activity of said monitored host;

one or more back-up hosts, each one of said back-up hosts comprising a Component Manager (CM), and at least one installed component;

wherein when a failure occurs in said monitored host, said one or more monitoring hosts detect said failure and start said at least one software component on at least one of said back-up hosts.

19. The network claimed in claim 18, wherein said monitoring hosts are the same as the back-up hosts.

20. The network claimed in claim 19, wherein said one or more monitoring hosts comprises a plurality of monitoring hosts, which divide said responsibility of starting and running said at least one component before effectively starting and running said at lest one component.

21. In a network of co-operating hosts, a method for re-assigning each software component of a monitored host to one or more monitoring hosts if said monitored host experiences a failure, the method comprising the steps of:

detecting a failure in the monitored host by a first monitoring host;

notifying a Central Information Repository (CIR) of the failure of the monitored host by said first monitoring host;

verifying, in said CIR, if other monitoring hosts than said first monitoring host are also responsible for monitoring an activity of said monitored host;

if other monitoring hosts are responsible for monitoring said activity of said monitored host, informing said other monitoring hosts of the failure of the monitored host;

obtaining, for each one of said first monitoring host and said other monitoring hosts, a list of software components run prior to said failure by the monitored host;

dividing a responsibility of re-starting individual components of said list among each one of said monitoring hosts; and

starting and activating each one of said individual components on a selected monitoring host according to said division of responsibility.

22. The method claimed in claim 21, wherein the step of dividing comprises, for a particular monitoring host, in a repetitive manner for each components of said list, the steps of:

selecting one component from said list;

obtaining at said particular monitoring host an actual host name entry for said particular component from said CIR;

comparing said actual host name entry with an identity of said monitored host; and

if a result of the step of comparing is a perfect match:

replacing said actual host name entry form said CIR with an identity of said particular monitoring host; and

deleting said one component from said list.

23. The method claimed in claim 22, wherein said steps are performed for each one of the components of said list.

24. The method claimed in claim 21, further comprising, subsequent to the step of starting, the steps of:

recovering the monitored host;

obtaining a list of a least one component said monitored host should be running from said CIR;

starting a first component identified in said list;

obtaining from said CIR an actual host name entry of said first component;

comparing by said monitored host the obtained actual host name entry of said first component with its own identity;

in case of a comparison mismatch, querying one of the monitored hosts corresponding to said actual host name entry for shutting down said first component;

shutting down said first component on said first monitoring hosts; and

updating said actual host name entry of said CIR with said identity of the monitored host.

25. A computer-operated software application for managing local software components running on a local host and for monitoring an activity of at least one networked monitored host, said application including a local Component Manager (CM) comprising:

means for detecting a failed monitored host;

means for obtaining an identity of at least one component run by said monitored host; and

means for starting and running a copy of said at least one component, wherein at least part of said copy is installed on said local host.

26. The computer-operated software application claimed in claim 25, wherein said CM further comprises:

means for dividing a responsibility of starting and running said at least one component between said local CM and a remote CM of another computer-operated software application running on another monitoring host responsible to monitor said activity of said monitored host.

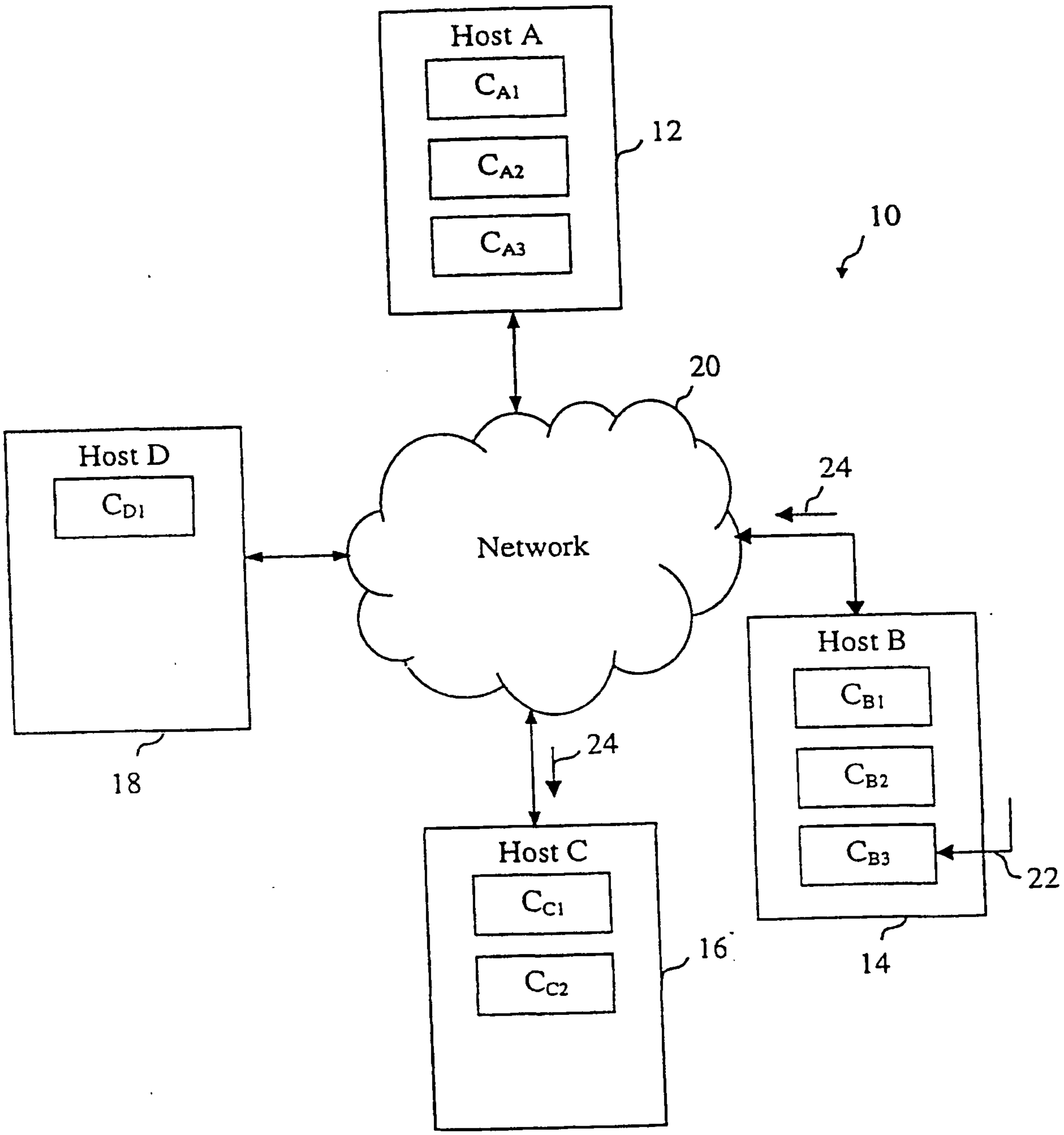

Figure 1.a (Prior Art)

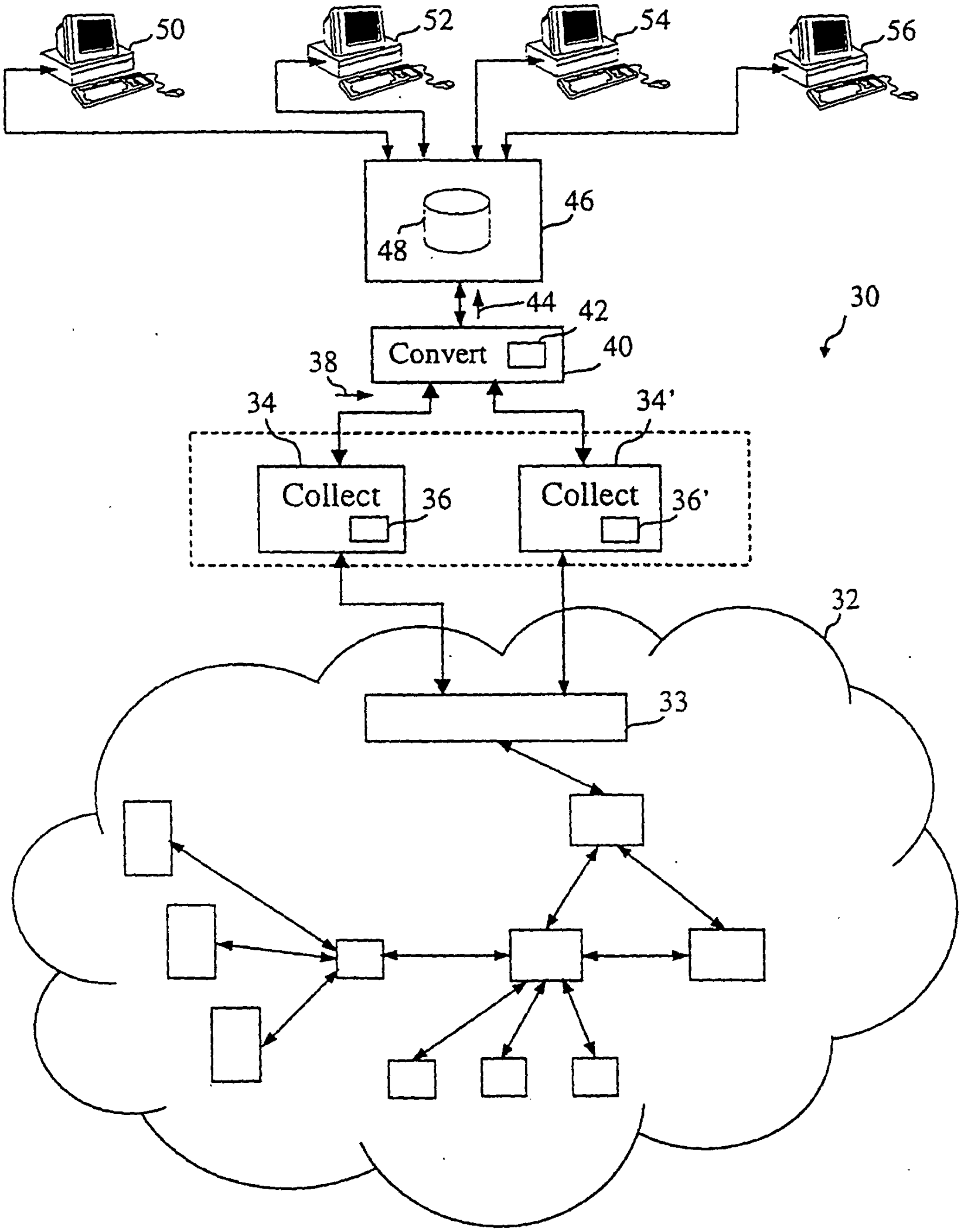

Figure 1.b (Prior Art)

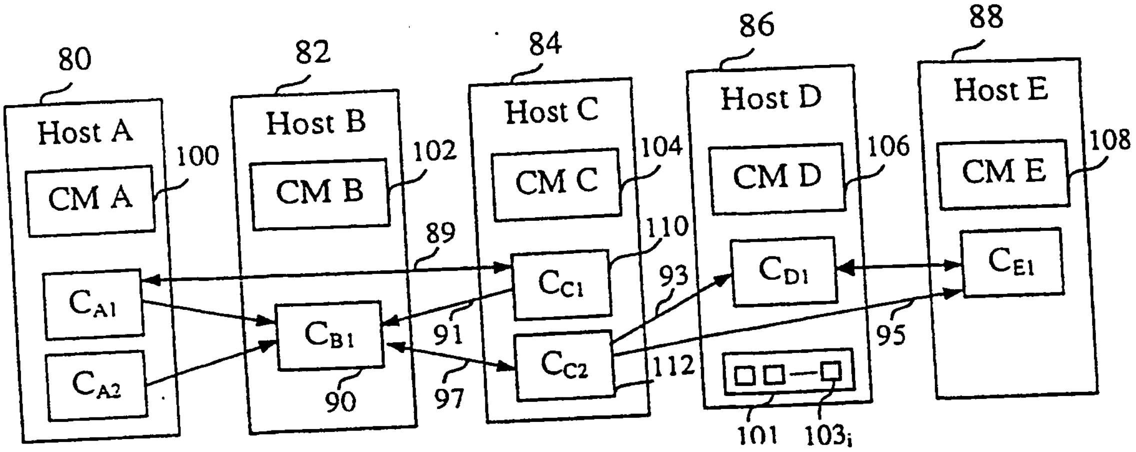

Figure 2.a

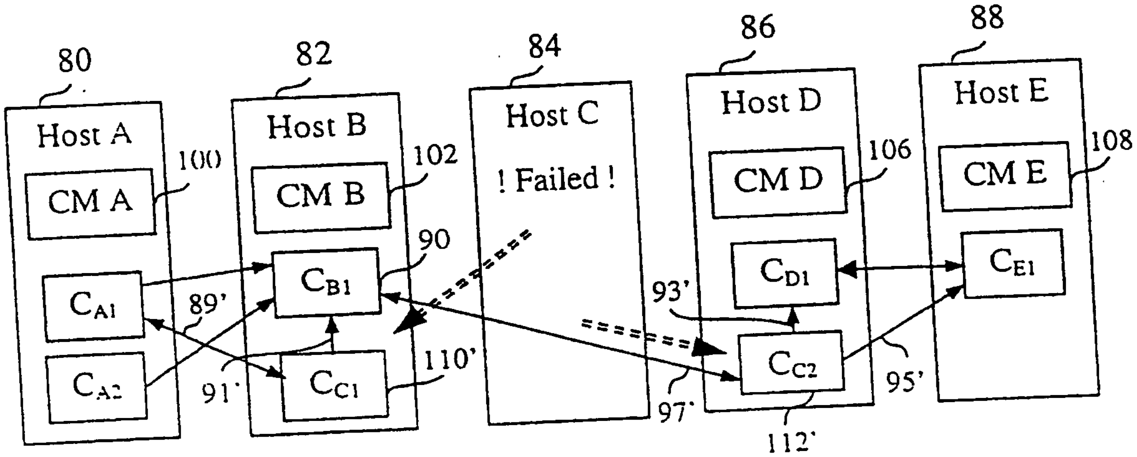

Figure 2.b

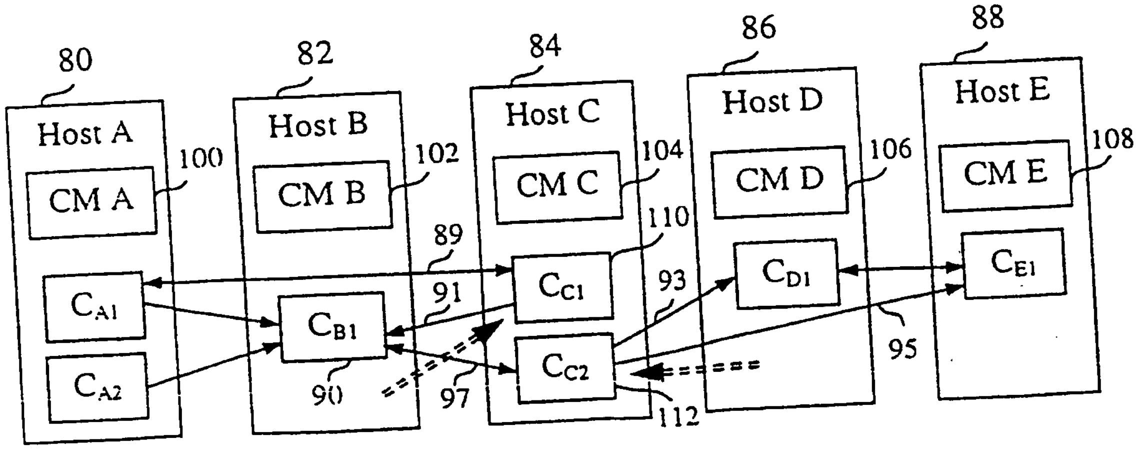

Figure 2.c

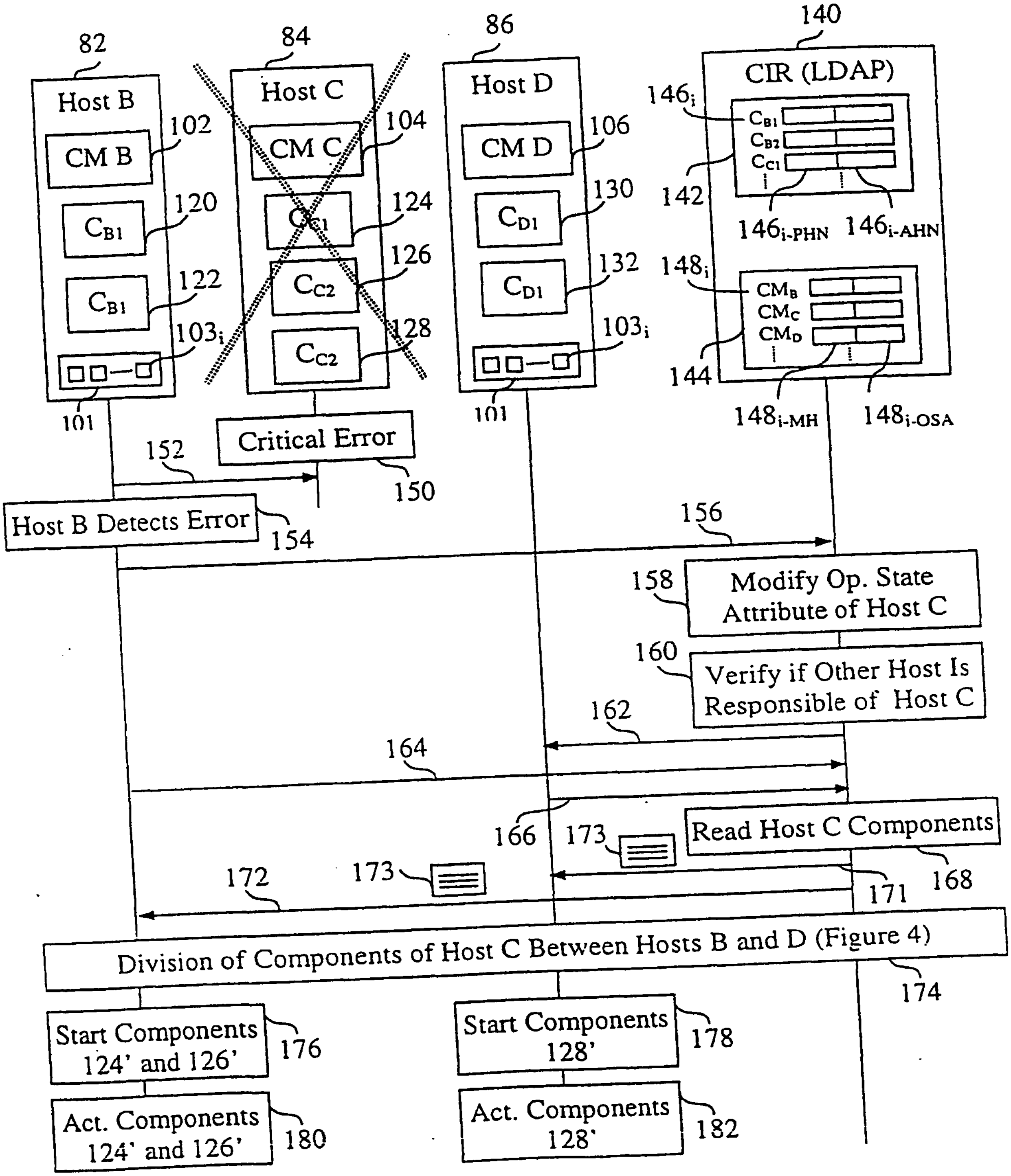
82
Host B
CM B
102
C_{B1}
120
C_{B1}
122
103_i
101
84
Host C
CM C
104
C_{C1}
124
C_{C2}
126
C_{C2}
128
86
Host D
CM D
106
C_{D1}
130
C_{D1}
132
103_i
101
140
CIR (LDAP)
146_i
C_{B1}
C_{B2}
C_{C1}
142
146_{i-PHN}
146_{i-AHN}
148_i
CM_B
CM_C
CM_D
144
148_{i-MH}
148_{i-OSA}
Critical Error
150
152
Host B Detects Error
154
156
Modify Op. State Attribute of Host C
158
Verify if Other Host Is Responsible of Host C
160
162
164
166
173
Read Host C Components
168
171
172
173
Division of Components of Host C Between Hosts B and D (Figure 4)
174
Start Components 124' and 126'
176
Start Components 128'
178
Act. Components 124' and 126'
180
Act. Components 128'
182

Figure 3

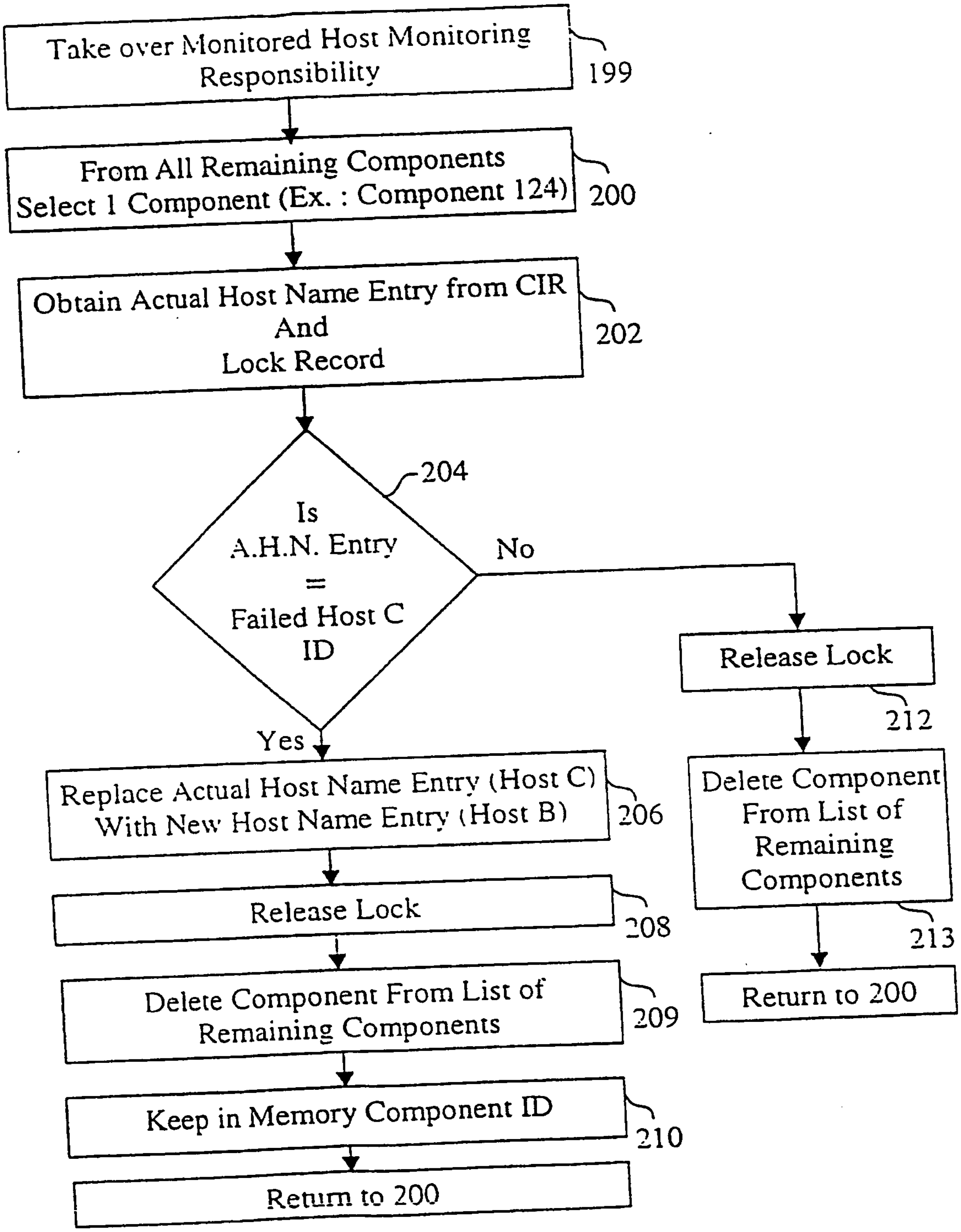
Take over Monitored Host Monitoring Responsibility
199
From All Remaining Components Select 1 Component (Ex. : Component 124)
200
Obtain Actual Host Name Entry from CIR And Lock Record
202
204
Is A.H.N. Entry = Failed Host C ID
No
Yes
Release Lock
212
Delete Component From List of Remaining Components
213
Return to 200
Replace Actual Host Name Entry (Host C) With New Host Name Entry (Host B)
206
Release Lock
208
Delete Component From List of Remaining Components
209
Keep in Memory Component ID
210
Return to 200

Figure 4

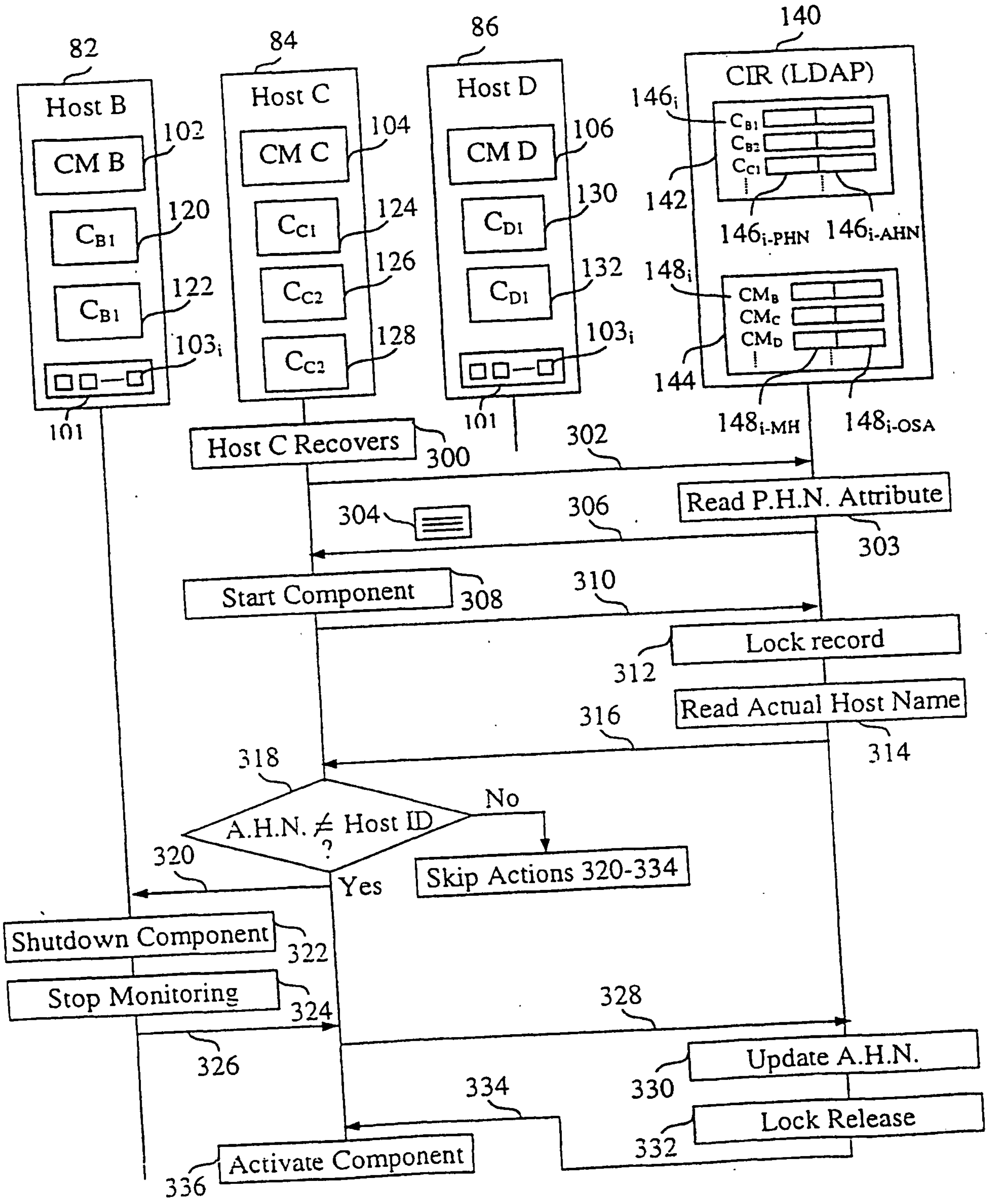
82
Host B
CM B
102
CB1
120
CB1
122
103i
101
84
Host C
CM C
104
CC1
124
CC2
126
CC2
128
86
Host D
CM D
106
CD1
130
CD1
132
103i
101
140
CIR (LDAP)
146i
CB1
CB2
CC1
142
146i-PHN
146i-AHN
148i
CMB
CMC
CMD
144
148i-MH
148i-OSA
Host C Recovers
300
302
Read P.H.N. Attribute
303
304
306
Start Component
308
310
Lock record
312
Read Actual Host Name
314
316
318
A.H.N. ≠ Host ID ?
No
Skip Actions 320-334
Yes
320
Shutdown Component
322
Stop Monitoring
324
326
328
Update A.H.N.
330
334
Lock Release
332
Activate Component
336

Figure 5